1-POP

Gaussian

```
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
### Test and CR for the mean of a multivariate Gaussian
### (one population)
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
library(car)
library(mvtnorm)
load("mcshapiro.test.RData")
data = read.table('stiff.dat', header=T)

# Testing
#    1.  formulate the test (and test gaussianity if needed)
#    2.  compute the test statistic
#    3a. set alpha and verify if the test statistic belongs to the rejection region
#    3b. compute the p-value of the test

# -----------------------------------------------------------------------------
# Test and CR_(1-alpha) for the mean of a multivariate Gaussian
# (one population)
# -----------------------------------------------------------------------------
#  H0: mu == mu0
#  H1: mu != mu0

n           = dim(data)[1]
p           = dim(data)[2]
data.mean   = sapply(data, mean)
data.cov    = cov(data)
data.invcov = solve(data.cov)

# Gaussianity
mcshapiro.test(data)

# if gaussianity is violated we can remove some outliers:
# d2 = matrix(mahalanobis(data, data.mean, data.cov))
# data = data[which(d2<7.5),]                              ## controlla che sia 7.5 ?
# n           = dim(data)[1]
# p           = dim(data)[2]
# data.mean   = sapply(data, mean)
# data.cov    = cov(data)
# data.invcov = solve(data.cov)

alpha = 0.01
mu0   = c(0,0)

# T2 statistics
data.T2 = n*(data.mean - mu0) %*% data.invcov %*% (data.mean - mu0)

# Radius of the ellipsoid
cfr.fisher = ((n-1)*p/(n-p))*qf(1-alpha, p, n-p)      # Fisher's quantile

# Test
data.T2 < cfr.fisher   # TRUE -> accept H0

# p-value
P = 1 - pf(data.T2*(n-p)/((n-1)*p), p, n-p)
P

# Plot
xx = seq(0,40,by=0.05)
plot(xx, df(xx*(n-p)/((n-1)*p),p,n-p), type="l", lwd=2, main='Density F(p,n-p)',
     xlab='x*(n-p)/((n-1)*p)', ylab='Density')
abline(h=0, v=data.T2*(n-p)/((n-1)*p), col=c('grey','red'), lwd=2, lty=c(2,1))

# -----------------------------------------------------------------------------
# IF p = 2     --     start
# -----------------------------------------------------------------------------
# Rejection region (centered in mu0)
# (outside the blue ellipse)
plot(data, asp=1)
ellipse(mu0, shape=data.cov/n, sqrt(cfr.fisher), col='blue', lty=2, center.pch=19)

# Sample mean on the plot: red point
points(data.mean[1], data.mean[2], pch=16, col='red', cex=1.5)

# Confidence region (centered in data.mean) level 100(1-alpha)%
# { x in R^2 s.t. n*(data.mean-x)' %*% (data.cov)^-1 %*% (data.mean-x) < cfr.fisher }
ellipse(data.mean, data.cov/n, sqrt(cfr.fisher), col='red', lty=2, lwd=2, center.cex=1)

# Confidence region with radius as the quantile of order 1-pval
ellipse(data.mean, data.cov/n, sqrt((n-1)*p/(n-p)*qf(1-as.numeric(P),p,n-p)),
        lty=1,col='dark grey',lwd=2)

# Comment
#   * If not specified the alpha (o se alpha è al limite, tipo tra 0.01 e 0.05)
#     prova con altri alpha
```

$H_0: \mu = \mu_0$
$H_1: \mu \neq \mu_0$

p = 2

```
#   * If the sample mean is inside the blue ellipse we accept H0
#     (meaning: the sample mean is not in the rejection region)
#   * If the mu0 (mean under H0) is in the confidence region of level 1-alpha
#     then we do not reject H0 at level alpha
#   * The confidece region of level 1-alpha contains all the mu0 that we would
#     accept at level alpha
#   * By def. the confidence region of level 1-alpha produces ellipsoidal regions that
#     contain the true mean 100(1-alpha)% of the times. If H0 is true (i.e mu0 is the
#     true mean), those ellipsoidal regions will contain mu0 100(1-alpha)% of the times

# ---------------------------------------------------------------------------------
# Intervals among directions
# ---------------------------------------------------------------------------------

# 1. Among the axes of the coordinates
T2 = cbind(inf = data.mean - sqrt(cfr.fisher*diag(data.cov)/n),
           center = data.mean,
           sup = data.mean + sqrt(cfr.fisher*diag(data.cov)/n))
T2

# Plot (il quadrato rosso sono i due CI)
plot(data, asp = 1, main='Confidence and rejection regions')
ellipse(mu0, shape=data.cov/n, sqrt(cfr.fisher), col='blue', lty=2, center.pch=16)
points(data.mean[1], data.mean[2], pch = 16, col='red', cex=1.5)
ellipse(data.mean, shape=data.cov/n, sqrt(cfr.fisher), col='red', lty=2, center.pch=16)
rect(T2[1,1],T2[2,1],T2[1,3],T2[2,3], border='red', lwd=2)

# 2. Among the worst direction
#    (direction along which the T2 statistics (univariate) is maximized)
data.T2  # maximum T2
worst = data.invcov %*% (data.mean - mu0)
worst = worst/sqrt(sum(worst^2))
worst
theta.worst = atan(worst[2]/worst[1])+pi
theta.worst
IC.worst  <- c(data.mean %*% worst - sqrt(cfr.fisher*(t(worst)%*%data.cov%*%worst)/n),
               data.mean %*% worst,
               data.mean %*% worst + sqrt(cfr.fisher*(t(worst)%*%data.cov%*%worst)/n) )
IC.worst

# Projecting mu0 on the worst direction
mu0 %*% worst

(IC.worst[1] < mu0%*%worst) & (mu0%*%worst < IC.worst[2])

# Extremes of IC.worst in the coordinate system (x,y):
x.min     = IC.worst[1]*worst
x.max     = IC.worst[3]*worst
m1.ort    = -worst[1]/worst[2]
q.min.ort = x.min[2] - m1.ort*x.min[1]
q.max.ort = x.max[2] - m1.ort*x.max[1]
abline(q.min.ort, m1.ort, col='forestgreen', lty=2,lwd=1)
abline(q.max.ort, m1.ort, col='forestgreen', lty=2,lwd=1)
m1 = worst[2]/worst[1] # worst direction
abline(0, m1, col='grey35')
segments(x.min[1],x.min[2],x.max[1],x.max[2],lty=1,lwd=2, col='forestgreen')

# Comment
# Se rifiutamo il test globale (H0), anche se i singoli IC contengono (rispettivamente)
# le due medie non cambia nulla, non è una contraddizione. Significa che stiamo
# rifiutando H0 su almeno una direzione, non necessariamente le direzioni delle coord.

# Bonferroni con generico k
k     = p
cfr.t = qt(1-alpha/(2*k), n-1)
Bf    = cbind(inf = data.mean - cfr.t*sqrt(diag(data.cov)/n),
              center = data.mean,
              sup = data.mean + cfr.t*sqrt(diag(data.cov)/n))
Bf
rect(Bf[1,1],Bf[2,1],Bf[1,3],Bf[2,3], border='orange', lwd=2)
legend('topleft', c('Rej. Reg.', 'Conf. Reg','T2-sim', 'Bonferroni'),
       col=c('blue','red','red','orange'),lty=c(2,2,1,1),lwd=2)

# Comment
# (Caso di rifiuto H0 globale ma entrambi i componenti della media rientrano dentro
# i signoli intervalli di confidenza)
# Può darsi che anche con la correzione di Bonferroni, i singoli CI accettino

# ---------------------------------------------------------------------------------
# IF p = 2    --    end
# ---------------------------------------------------------------------------------

# ---------------------------------------------------------------------------------
# IF p > 2    --    start
# ---------------------------------------------------------------------------------
# Confidence regions for the mean of level 100(1-alpha)%
#   * we want the CR for the mean (ellipsoidal region)
```

p = 2

p > 2

Bonferroni with generic k on σ^2

k = p

s2 = diag(data.cov)

chi1 = qchisq(1-α/(2*k), n-1

chi2 = qchisq(α/(2*k), n-1

Bf_sig = cbind(inf = (n-1)*s2/ch

sup = (n-1)*S2/ch

```
#     { x in R^4 t.c. n*(data.mean-x)' %*% data.invcov %*% (data.mean-m) < cfr.fisher }
#   * characterization of the region: centre, direction of the principal axes,
#     length of the axes

# Centre
data.mean

# Direction of the semi-axes of the ellipse
r = sqrt(cfr.fisher)
r*sqrt(eigen(data.cov/n)$values)

# We plot the projections of the ellipsoid in some directions of interest
# (e.g the x and y coordinates)
# We plot simultaneous T2 confidence intervals in each direction of interest
# (with global coverage alpha)
T2 = cbind(inf = data.mean - sqrt(cfr.fisher*diag(data.cov)/n),
           center = data.mean,
           sup = data.mean + sqrt(cfr.fisher*diag(data.cov)/n))
T2

matplot(1:p, 1:p, pch='', ylim=range(T2), xlab='Variables', ylab='T2 for a component',
        main='Simultaneous T2 conf. int. for the components')
for(i in 1:p) segments(i,T2[i,1],i,T2[i,3],lwd=3,col=i)
points(1:p, T2[,2], pch=16, col=1:4)

# Is mu0 inside the rectangular region?
points(1:p, mu0, lwd=3, col='orange', pch=19)

# Bonferroni
# (with global level 100(1-alpha)%)
k     = p
cfr.t = qt(1 - alpha/(k*2), n-1)
Bf    = cbind(inf = data.mean - cfr.t*sqrt(diag(data.cov)/n),
              center = data.mean,
              sup = data.mean + cfr.t*sqrt(diag(data.cov)/n))
Bf
matplot(1:p, 1:p, pch='', ylim=range(T2), xlab='Variables',
        ylab='Confidence intervals along a component',main='Confidence intervals')

for(i in 1:p) segments(i,T2[i,1],i,T2[i,3],lwd=2,col='grey35', lty=3)
points(1:p, T2[,1], pch='-', col='grey35')
points(1:p, T2[,3], pch='-', col='grey35')

for(i in 1:p) segments(i,Bf[i,1],i,Bf[i,3],lwd=2,col=i)
points(1:p, Bf[,2], pch=16, col=1:p)
points(1:p, Bf[,1], pch='-', col=1:p)
points(1:p, Bf[,3], pch='-', col=1:p)

# Is mu0 inside the Bonferroni confidence region?
# we add it to the plot
points(1:p, mu0, lwd=3, col='orange', pch=19)

# ---
# PLOT OF ONE-AT-TIME IC AND BONFERRONI TOGETHER (p=3)
# in questo caso il numero di componenti di mu0 è 3
# ---
matplot(t(matrix(1:3,3,3)),t(Bf), type='b',pch='',
        xlim=c(0,4),xlab='',ylab='', main='Confidence intervals')
segments(matrix(1:3,3,1),Bf[,1],matrix(1:3,3,1),Bf[,3], col='orange', lwd=2)
points(1:3, Bf[,2], col='orange', pch=16)
points(1:3+.05, mu0, col='black', pch=16)
segments(matrix(1:3+.1,3,1),T2[,1],matrix(1:3+.1,3,1),T2[,3], col='blue', lwd=2)
points(1:3+.1,T2[,2], col='blue', pch=16)

# ------------------------------------------------------------------------------------
# IF p > 2    --    end
# ------------------------------------------------------------------------------------

# ------------------------------------------------------------------------------------
# Asymptotic test on the mean
# (no gaussianity of the data but n must be high)
# ------------------------------------------------------------------------------------
n           = dim(data)[1]
p           = dim(data)[2]
data.mean   = sapply(data, mean)
data.cov    = cov(data)
data.invcov = solve(data.cov)

alpha = 0.01
mu0   = c(1900, 1700)

# T2 statistics
data.T2A = n*(data.mean - mu0) %*% data.invcov %*% (data.mean - mu0)

# Radius of the ellipsoid
cfr.chisq = qchisq(1-alpha, p)      # Chi-square's quantile
```

p > 2

asymptotic

asymptotic

```
# Test
data.T2A < cfr.chisq   # TRUE -> accept H0

# p-value
PA = 1 - pchisq(data.T2A, p)
PA

# Plot (controlla!)
curve(dchisq(x, df = p), from = 0, to = 40, main = 'Chi-Square Distribution',
      ylab = 'Density', lwd = 2)
abline(h=0, v=data.T2A, col=c('grey','red'), lwd=2, lty=c(2,1))

# ------------------------------------------------------------------------------------
# IF p = 2    --    start
# ------------------------------------------------------------------------------------
# Comparison of the rejection regions
# (Fisher's vs. Chi-square's)
# ------------------------------------------------------------------------------------
plot(data, asp = 1,main='Comparison rejection regions')
ellipse(mu0, shape=data.cov/n, sqrt(cfr.fisher), col = 'blue',
        lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
ellipse(mu0, data.cov/n, sqrt(cfr.chisq), col = 'lightblue',
        lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
points(mu0[1], mu0[2], pch = 4, cex = 1.5, lwd = 2, col ='lightblue')
legend('topleft', c('Exact', 'Asymptotic'),col=c('blue','lightblue'),lty=c(1),lwd=2)

# Comparison of the confidence regions
plot(data, asp = 1,main='Comparison of confidence regions')
ellipse(data.mean, data.cov/n, sqrt(cfr.fisher), col = 'red',
        lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
ellipse(data.mean, data.cov/n, sqrt(cfr.chisq), col = 'orange',
        lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
points(data.mean[1], data.mean[2], pch = 4, cex = 1.5, lwd = 2, col ='orange')
legend('topleft', c('Exact', 'Asymptotic'),col=c('red','orange'),lty=c(1),lwd=2)
# ------------------------------------------------------------------------------------
# IF p = 2    --    end
# ------------------------------------------------------------------------------------
```

Paired

Hide | Code

```
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
### Paired Gaussian data: test for the mean
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
library(car)
load("mcshapiro.test.RData")
data = read.table('effluent.dat')

# -------------------------------------------------------------------------------
# Impostazione del vettore (multidimensionale) media
# -------------------------------------------------------------------------------
# Consideriamo un dataset con k colonne che verrano messe in coppia a due a due
#       ---------------------------
#       |     | V1 | V2 | V3 | V4 |
#       ---------------------------
#       |  1  |    |    |    |    |
#       |  2  |    |    |    |    |
#       |     |    |    |    |    |
# e lavoriamo su una media multidimensionale formata da k/2 componenti:
# tipo qui avremo mu = (V1-V3, V2-V4) e vorremo vedere se sarà 0
#                    = (  X1 ,   X2 )

pairs(data, pch=19)

D = data.frame(X1 = data$V1 - data$V3, X2 = data$V2 - data$V4)
D

# Plot
plot(D, asp=1, pch=19, main='dataset of differences')
abline(h=0, v=0, col='grey')
points(0,0, pch=19, col='grey')
# (do we have enough evidence to say that the grey point is the mean?)

data = D
# -------------------------------------------------------------------------------
# DA QUI SI FA PARTIRE LO SCRIPT
# '(1 pop. gauss) test and CR for the mean.r'
# -------------------------------------------------------------------------------
```

Hide

Repeated

```
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
### Repeated measures (q measures)
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
library(car)
load("mcshapiro.test.RData")
data = read.table ('pressure.txt', col.names=c('h.0','h.8','h.16','h.24'))

# -------------------------------------------------------------------------------
# Checking assumptions and explorating
# -------------------------------------------------------------------------------
mcshapiro.test(data)

matplot(t(data), type='l')
# Per il matplot i dati devono essere tipo:
#       ---------------------------------------
#       |     | meas.1 | meas.2 | .. | meas.q |
#       ---------------------------------------
#       |  1  |   ..   |   ..   | .. |   ..   |         dove le righe sono independent
#       |  2  |   ..   |   ..   | .. |   ..   |         statistical units
#       |     |   ..   |   ..   | .. |   ..   |

n = dim(data)[1]
q = dim(data)[2]

M = sapply(data, mean)
M

S = cov(data)
S

# -------------------------------------------------------------------------------
# Contrast matrix
# -------------------------------------------------------------------------------
C = matrix(c(-1, 1, 0, 0,
             -1, 0, 1, 0,
             -1, 0, 0, 1), 3, 4, byrow=T)
C

# -------------------------------------------------------------------------------
# Testing
# H0: C%*%mu == 0
# H1: C%*%mu != 0
# -------------------------------------------------------------------------------

data.mean   = C %*% M
data.cov    = C %*% S %*% t(C)
data.invcov = solve(data.cov)
mu0         = seq(0,0, length = q-1)
alpha       = 0.05
p           = q-1
data.T2     = n * t(data.mean - mu0) %*% data.invcov %*% (data.mean - mu0)

# -------------------------------------------------------------------------------
# DA QUI SI FA PARTIRE LO SCRIPT (ATTENZIONE!!)
# '(1 pop. gauss) test and CR for the mean.r'
# da "Radius of ellipsoid" in poi
# -------------------------------------------------------------------------------

# INTERPRETAZIONE
# Caso        :  1 misurazione (x0) prima della drug, 3 misurazioni (x1,x2,x3) post drug
#
# Richiesta_1 : "Perform a test at level 5% to prove that the drug has influence"
#               (Attenzione: solo test! Gli CI sicuro verrano chiesti dopo)
# Procedimento: (1) contrast matrix che paragona (x1-x0, x2-x0, x3-x0)
#               (2) contrast matrix che paragona (x1-x0, x2-x1, x3-x2)
#
# Richiesta_2 : "Highlight the effect of the drug on the blood pressure"
# Procedimento: alcuni CI per vedere se la <variabile> cambia (e.g. pressione)
#
# RIchiesta_3 : "The drug decreases the pressure of two units w.r.t. the baseline at
#                both times x1 and x2, and its effect vanishes at x3"
# Procedimento: C = matrix(c(-1, 1, 0, 0,
#                            -1, 0, 1, 0,
#                            -1, 0, 0, 1), 3, 4, byrow=T)
#               mu0 = c(2,2,0)
```

$$\begin{cases} H_0: C\underline{\mu} = \underline{0} \\ H_1: C\underline{\mu} \neq \underline{0} \end{cases}$$

2- POP

Hide Code

```
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
### Test for two idependent Gaussian populations
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
library(car)
load("mcshapiro.test.RData")

data1 = read.table("lovingalmonds.txt", head=T)
data2 = read.table("hatingalmonds.txt", head=T)

# -------------------------------------------------------------------------------
# Checking assumptions and explorating
# -------------------------------------------------------------------------------

n1 = dim(data1)[1]
n2 = dim(data2)[1]
p  = dim(data1)[2]

data1.mean = sapply(data1, mean)
data2.mean = sapply(data2, mean)
data1.cov  = cov(data1)
data2.cov  = cov(data2)
Sp         = ((n1-1)*data1.cov + (n2-1)*data2.cov)/(n1+n2-2)
Sp.inv     = solve(Sp)

# Comparison of the matrices
list( S1=data1.cov, S2=data2.cov, Spooled=Sp)

# -------------------------------------------------------------------------------
# Test for the mean of a multivariate (p>1) Gaussian
# (two population)
# -------------------------------------------------------------------------------
#  H0: mu1 == mu2
#  H1: mu1 != mu2

alpha  = 0.01
delta0 = c(0,0)

# Test statistic
T2 = n1*n2/(n1+n2) * (data1.mean-data2.mean-delta0) %*% Sp.inv %*%
  (data1.mean-data2.mean-delta0)

# Radius of the ellpisoid
cfr.fisher = (p*(n1+n2-2)/(n1+n2+1-p))*qf(1-alpha, p, n1+n2-1-p)

# Test
T2 < cfr.fisher

# p-value
P = 1 - pf(T2/(p*(n1+n2-2)/(n1+n2-1-p)), p, n1+n2-1-p)
P

# Simultaneous T2 intervals
IC.T2.p1 = c(data1.mean[1]-data2.mean[1]-sqrt(cfr.fisher*Sp[1,1]*(1/n1+1/n2)),
             data1.mean[1]-data2.mean[1]+sqrt(cfr.fisher*Sp[1,1]*(1/n1+1/n2)))
IC.T2.p2 = c(data1.mean[2]-data2.mean[2]-sqrt(cfr.fisher*Sp[2,2]*(1/n1+1/n2)),
             data1.mean[2]-data2.mean[2]+sqrt(cfr.fisher*Sp[2,2]*(1/n1+1/n2)))
IC.T2 = rbind(IC.T2.p1, IC.T2.p2)
dimnames(IC.T2)[[2]] <- c('inf','sup')
IC.T2

# Comment
# Do the interals (separately) contains the value 0?
```